Foundation of Cryptography, Lecture 5
MACs and Signatures

Iftach Haitner, Tel Aviv University

Tel Aviv University.

April 30, 2013

Section 1

Message Authentication Code (MAC)

Message Authentication Code (MAC)

Definition 1 (MAC)

A trippet of PPT's $(\mathsf{Gen}, \mathsf{Mac}, \mathsf{Vrfy})$ such that

1. $\mathsf{Gen}(1^n)$ outputs a key $k \in \{0,1\}^*$
2. $\mathsf{Mac}(k, m)$ outputs a "tag" t
3. $\mathsf{Vrfy}(k, m, t)$ output 1 (YES) or 0 (NO)

Consistency: $\mathsf{Vrfy}_k(m, t) = 1$
$\forall k \in \mathsf{Supp}(\mathsf{Gen}(1^n))$, $m \in \{0,1\}^n$ and $t = \mathsf{Mac}_k(m)$

Message Authentication Code (MAC)

Definition 1 (MAC)

A trippet of PPT's $(\mathsf{Gen}, \mathsf{Mac}, \mathsf{Vrfy})$ such that

1. $\mathsf{Gen}(1^n)$ outputs a key $k \in \{0,1\}^*$
2. $\mathsf{Mac}(k, m)$ outputs a "tag" t
3. $\mathsf{Vrfy}(k, m, t)$ output 1 (YES) or 0 (NO)

Consistency: $\mathsf{Vrfy}_k(m, t) = 1$
$\forall k \in \mathsf{Supp}(\mathsf{Gen}(1^n))$, $m \in \{0,1\}^n$ and $t = \mathsf{Mac}_k(m)$

Definition 2 (Existential unforgability)

A MAC $(\mathsf{Gen}, \mathsf{Mac}, \mathsf{Vrfy})$ is existential unforgeable (EU), if $\forall$ PPT A:

$$\Pr\Big[k \leftarrow \mathsf{Gen}(1^n); (m, t) \leftarrow \mathsf{A}^{\mathsf{Mac}_k, \mathsf{Vrfy}_k}(1^n): \\ \mathsf{Vrfy}_k(m, t) = 1 \wedge \mathsf{Mac}_k \text{ was not asked on } m\Big] = \mathsf{neg}(n)$$

Definition of MAC cont.

- “Private key" definition

Definition of MAC cont.

- “Private key" definition
- Security definition too strong? Any message? Use of Verifier?

Definition of MAC cont.

- “Private key" definition
- Security definition too strong? Any message? Use of Verifier?

Definition of MAC cont.

- “Private key" definition
- Security definition too strong? Any message? Use of Verifier?
- “Replay attacks"

Definition of MAC cont.

- “Private key" definition
- Security definition too strong? Any message? Use of Verifier?
- “Replay attacks"
- Strong existential unforgeable MACS (for short, strong MAC: infeasible to generate new valid tag (even for message for which a MAC was asked)

Length-restricted MACs

Definition 3 (Length-restricted MAC)

Same as in Definition 1, but for $k \in \mathrm{Supp}(G(1^n))$, Mac_k and Vrfy_k only accept messages of length n.

Bounded-query MACs

Definition 4 (ℓ-time MAC)

A MAC scheme is existential unforgeable against ℓ queries (for short, ℓ-time MAC), if it is existential unforgeable as in Definition 2, but A can only make ℓ queries.

Section 2

Constructions

Zero-time MAC

Construction 5 (Zero-time MAC)

- $\mathsf{Gen}(1^n)$: outputs $k \leftarrow \{0,1\}^n$
- $\mathsf{Mac}_k(m) = k$
- $\mathsf{Vrfy}_k(m,t) = 1$, iff $t = k$

Zero-time MAC

Construction 5 (Zero-time MAC)

- $\text{Gen}(1^n)$: outputs $k \leftarrow \{0,1\}^n$
- $\text{Mac}_k(m) = k$
- $\text{Vrfy}_k(m,t) = 1$, iff $t = k$

Claim 6

The above scheme is zero-time MAC

Zero-time MAC

Construction 5 (Zero-time MAC)

- $\mathsf{Gen}(1^n)$: outputs $k \leftarrow \{0,1\}^n$
- $\mathsf{Mac}_k(m) = k$
- $\mathsf{Vrfy}_k(m,t) = 1$, iff $t = k$

Claim 6

The above scheme is zero-time MAC

Does it remind you something?

ℓ-wise Independent Hash

Definition 7 (ℓ-wise independent)

A function family $\mathcal{H}$ from $\{0,1\}^n$ to $\{0,1\}^m$ is ℓ-wise independent, where $\ell \in \mathbb{N}$, if for every distinct $x_1, \ldots, x_\ell \in \{0,1\}^n$ and every $y_1, \ldots, y_\ell \in \{0,1\}^m$, it holds that $\Pr_{h \leftarrow \mathcal{H}}[h(x_1) = y_1 \wedge \cdots \wedge h(x_\ell) = y_\ell] = 2^{-\ell m}$.

ℓ-times, Restricted Length, MAC

Construction 8 (ℓ-time MAC)

Let $\mathcal{H} = \{\mathcal{H}_n \colon \{0,1\}^n \mapsto \{0,1\}^n\}$ be an efficient $(\ell+1)$-wise independent function family.

- $\mathsf{Gen}(1^n)$: outputs $h \leftarrow \mathcal{H}_n$
- $\mathsf{Mac}(h, m) = h(m)$
- $\mathsf{Vrfy}(h, m, t) = 1$, iff $t = h(m)$

ℓ-times, Restricted Length, MAC

Construction 8 (ℓ-time MAC)

Let $\mathcal{H} = \{\mathcal{H}_n \colon \{0,1\}^n \mapsto \{0,1\}^n\}$ be an efficient $(\ell+1)$-wise independent function family.

- $\mathsf{Gen}(1^n)$: outputs $h \leftarrow \mathcal{H}_n$
- $\mathsf{Mac}(h, m) = h(m)$
- $\mathsf{Vrfy}(h, m, t) = 1$, iff $t = h(m)$

Claim 9

The above scheme is a length-restricted, ℓ-time MAC

ℓ-times, Restricted Length, MAC

Construction 8 (ℓ-time MAC)

Let $\mathcal{H} = \{\mathcal{H}_n \colon \{0,1\}^n \mapsto \{0,1\}^n\}$ be an efficient $(\ell + 1)$-wise independent function family.

- $\mathsf{Gen}(1^n)$: outputs $h \leftarrow \mathcal{H}_n$
- $\mathsf{Mac}(h, m) = h(m)$
- $\mathsf{Vrfy}(h, m, t) = 1$, iff $t = h(m)$

Claim 9

The above scheme is a length-restricted, ℓ-time MAC

Proof: ?

OWF $\implies$ Existential Unforgeable MAC

Construction 10

Same as Construction 8, but uses function $\mathcal{F} = \{\mathcal{F}_n \colon \{0,1\}^n \mapsto \{0,1\}^n\}$ instead of $\mathcal{H}$.

OWF $\Longrightarrow$ Existential Unforgeable MAC

Construction 10

Same as Construction 8, but uses function $\mathcal{F} = \{\mathcal{F}_n \colon \{0,1\}^n \mapsto \{0,1\}^n\}$ instead of $\mathcal{H}$.

Claim 11

Assuming that $\mathcal{F}$ is a PRF, then Construction 10 is an existential unforgeable MAC.

OWF $\implies$ Existential Unforgeable MAC

Construction 10

Same as Construction 8, but uses function $\mathcal{F} = \{\mathcal{F}_n \colon \{0,1\}^n \mapsto \{0,1\}^n\}$ instead of $\mathcal{H}$.

Claim 11

Assuming that $\mathcal{F}$ is a PRF, then Construction 10 is an existential unforgeable MAC.

Proof:

OWF $\implies$ Existential Unforgeable MAC

Construction 10

Same as Construction 8, but uses function
$\mathcal{F} = \{\mathcal{F}_n \colon \{0,1\}^n \mapsto \{0,1\}^n\}$ instead of $\mathcal{H}$.

Claim 11

Assuming that $\mathcal{F}$ is a PRF, then Construction 10 is an existential unforgeable MAC.

Proof: Easy to prove if $\mathcal{F}$ is a family of random functions. Hence, also holds in case $\mathcal{F}$ is a PRF.$\square$

Collision Resistant Hash Family

Definition 12 (collision resistant hash family (CRH))

A function family $\mathcal{H} = \{\mathcal{H}_n \colon \{0,1\}^* \mapsto \{0,1\}^n\}$ is collision resistant, if

$$\Pr[h \leftarrow \mathcal{H}_n, (x, x') \leftarrow \mathsf{A}(1^n, h) \colon x \neq x' \in \{0,1\}^* \\ \wedge h(x) = h(x')] = \operatorname{neg}(n)$$

for any PPT A.

Collision Resistant Hash Family

Definition 12 (collision resistant hash family (CRH))

A function family $\mathcal{H} = \{\mathcal{H}_n \colon \{0,1\}^* \mapsto \{0,1\}^n\}$ is collision resistant, if

$$\Pr[h \leftarrow \mathcal{H}_n, (x, x') \leftarrow \mathsf{A}(1^n, h) \colon x \neq x' \in \{0,1\}^* \\ \wedge h(x) = h(x')] = \operatorname{neg}(n)$$

for any PPT A.

- Not known to be implied by OWF

Length restricted MAC $\implies$ MAC

Construction 13 (Length restricted MAC $\implies$ MAC)

Let $(\mathsf{Gen}, \mathsf{Mac}, \mathsf{Vrfy})$ be a length-restricted MAC, and let $\mathcal{H} = \{\mathcal{H}_n \colon \{0,1\}^* \mapsto \{0,1\}^n\}$ be an efficient function family.

- $\mathsf{Gen}'(1^n)$: $k \leftarrow \mathsf{Gen}(1^n)$, $h \leftarrow \mathcal{H}_n$. Set $k' = (k, h)$
- $\mathsf{Mac}'_{k,h}(m) = \mathsf{Mac}_k(h(m))$
- $\mathsf{Vrfy}'_{k,h}(t, m) = \mathsf{Vrfy}_k(t, h(m))$

Length restricted MAC $\implies$ MAC

Construction 13 (Length restricted MAC $\implies$ MAC)

Let $(\mathsf{Gen}, \mathsf{Mac}, \mathsf{Vrfy})$ be a length-restricted MAC, and let $\mathcal{H} = \{\mathcal{H}_n \colon \{0,1\}^* \mapsto \{0,1\}^n\}$ be an efficient function family.

- $\mathsf{Gen}'(1^n)$: $k \leftarrow \mathsf{Gen}(1^n)$, $h \leftarrow \mathcal{H}_n$. Set $k' = (k, h)$
- $\mathsf{Mac}'_{k,h}(m) = \mathsf{Mac}_k(h(m))$
- $\mathsf{Vrfy}'_{k,h}(t, m) = \mathsf{Vrfy}_k(t, h(m))$

Claim 14

Assume $\mathcal{H}$ is an efficient collision-resistant family and $(\mathsf{Gen}, \mathsf{Mac}, \mathsf{Vrfy})$ is existential unforgeable, then $(\mathsf{Gen}', \mathsf{Mac}', \mathsf{Vrfy}')$ is existential unforgeable MAC.

Length restricted MAC $\implies$ MAC

Construction 13 (Length restricted MAC $\implies$ MAC)

Let $(\mathsf{Gen}, \mathsf{Mac}, \mathsf{Vrfy})$ be a length-restricted MAC, and let $\mathcal{H} = \{\mathcal{H}_n \colon \{0,1\}^* \mapsto \{0,1\}^n\}$ be an efficient function family.

- $\mathsf{Gen}'(1^n)$: $k \leftarrow \mathsf{Gen}(1^n)$, $h \leftarrow \mathcal{H}_n$. Set $k' = (k, h)$
- $\mathsf{Mac}'_{k,h}(m) = \mathsf{Mac}_k(h(m))$
- $\mathsf{Vrfy}'_{k,h}(t, m) = \mathsf{Vrfy}_k(t, h(m))$

Claim 14

Assume $\mathcal{H}$ is an efficient collision-resistant family and $(\mathsf{Gen}, \mathsf{Mac}, \mathsf{Vrfy})$ is existential unforgeable, then $(\mathsf{Gen}', \mathsf{Mac}', \mathsf{Vrfy}')$ is existential unforgeable MAC.

Proof: ?

Section 3

Signature Schemes

Defining Signature Schemes

Definition 15 (Signature schemes)

A trippet of PPT's $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ such that

1. $\mathsf{Gen}(1^n)$ outputs a pair of keys $(s, v) \in \{0,1\}^* \times \{0,1\}^*$
2. $\mathsf{Sign}(s, m)$ outputs a “signature" $\sigma \in \{0,1\}^*$
3. $\mathsf{Vrfy}(v, m, \sigma)$ outputs 1 (YES) or 0 (NO)

Consistency: $\mathsf{Vrfy}_v(m, \sigma) = 1$ for any $(s, v) \in \mathsf{Supp}(\mathsf{Gen}(1^n))$, $m \in \{0,1\}^*$ and $\sigma \in \mathsf{Supp}(\mathsf{Sign}_s(m))$

Defining Signature Schemes

Definition 15 (Signature schemes)

A trippet of PPT's $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ such that

1. $\mathsf{Gen}(1^n)$ outputs a pair of keys $(s, v) \in \{0,1\}^* \times \{0,1\}^*$
2. $\mathsf{Sign}(s, m)$ outputs a "signature" $\sigma \in \{0,1\}^*$
3. $\mathsf{Vrfy}(v, m, \sigma)$ outputs 1 (YES) or 0 (NO)

Consistency: $\mathsf{Vrfy}_v(m, \sigma) = 1$ for any $(s, v) \in \mathsf{Supp}(\mathsf{Gen}(1^n))$, $m \in \{0,1\}^*$ and $\sigma \in \mathsf{Supp}(\mathsf{Sign}_s(m))$

Definition 16 (Existential unforgability)

A signature scheme is existential unforgeable (EU), if $\forall$ PPT A

$$\Pr\Big[(s, v) \leftarrow \mathsf{Gen}(1^n); (m, \sigma) \leftarrow \mathsf{A}^{\mathsf{Sign}_s}(1^n, v) : \\ \mathsf{Vrfy}_v(m, \sigma) = 1 \land \mathsf{Sign}_s \text{ was not asked on } m\Big] = \mathsf{neg}(n)$$

Defining Signature Schemes cont.

- Signature $\implies$ MAC

Defining Signature Schemes cont.

- Signature $\Longrightarrow$ MAC
- “Harder" to construct than MACs: (even restricted forms) require OWF

Defining Signature Schemes cont.

- Signature $\implies$ MAC
- “Harder" to construct than MACs: (even restricted forms) require OWF
- Oracle access to Vrfy is not given

Defining Signature Schemes cont.

- Signature $\implies$ MAC
- "Harder" to construct than MACs: (even restricted forms) require OWF
- Oracle access to Vrfy is not given
- Strong existential unforgeable signatures (for short, strong signatures): infeasible to generate new valid signatures (even for message for which a signature was asked)

Defining Signature Schemes cont.

- Signature $\implies$ MAC
- “Harder" to construct than MACs: (even restricted forms) require OWF
- Oracle access to Vrfy is not given
- Strong existential unforgeable signatures (for short, strong signatures): infeasible to generate new valid signatures (even for message for which a signature was asked)

Defining Signature Schemes cont.

- Signature $\implies$ MAC
- “Harder" to construct than MACs: (even restricted forms) require OWF
- Oracle access to Vrfy is not given
- Strong existential unforgeable signatures (for short, strong signatures): infeasible to generate new valid signatures (even for message for which a signature was asked)

Theorem 17

OWFs imply strong existential unforgeable signatures.

Section 4

OWFs $\implies$ Signatures

Length-restricted signatures

Definition 18 (Length-restricted Signatures)

Same as in Definition 15, but for $(s, v) \in \mathrm{Supp}(G(1^n))$, Sign_s and Vrfy_v only accept messages of length n.

Bounded-query Signatures

Definition 19 (ℓ-time signatures)

A signature scheme is existential unforgeable against ℓ-query (for short, ℓ-time signature), if it is existential unforgeable as in Definition 16, but A can only ask for ℓ queries.

Bounded-query Signatures

Definition 19 (ℓ-time signatures)

A signature scheme is existential unforgeable against ℓ-query (for short, ℓ-time signature), if it is existential unforgeable as in Definition 16, but A can only ask for ℓ queries.

Claim 20

Assuming CRH exists, then length restricted, one-time signatures can be used to construct one-time signatures.

Bounded-query Signatures

Definition 19 (ℓ-time signatures)

A signature scheme is existential unforgeable against ℓ-query (for short, ℓ-time signature), if it is existential unforgeable as in Definition 16, but A can only ask for ℓ queries.

Claim 20

Assuming CRH exists, then length restricted, one-time signatures can be used to construct one-time signatures.

Proof?

Bounded-query Signatures

Definition 19 (ℓ-time signatures)

A signature scheme is existential unforgeable against ℓ-query (for short, ℓ-time signature), if it is existential unforgeable as in Definition 16, but A can only ask for ℓ queries.

Claim 20

Assuming CRH exists, then length restricted, one-time signatures can be used to construct one-time signatures.

Proof?

Proposition 21

Wlg, the signer of a one-time signature is deterministic

OWF $\implies$ Length Restricted, One Time Signature

Construction 22 (length-restricted, one-time signature)

Let $f \colon \{0,1\}^n \mapsto \{0,1\}^n$.

1. $\mathsf{Gen}(1^n)$:
   1. $s_1^0, s_1^1, \ldots, s_n^0, s_n^1 \leftarrow \{0,1\}^n$,
   2. $s = \left(s_1^0, s_1^1, \ldots, s_n^0, s_n^1\right)$
   3. $v = \left(v_1^0 = f(s_1^0), v_1^1 = f(s_1^1), \ldots, v_n^0 = f(s_n^0), v_n^1 = f(s_n^1)\right)$
2. $\mathsf{Sign}(s, m)$: $\sigma = (s_1^{m_1}, \ldots, s_n^{m_n})$
3. $\mathsf{Vrfy}(v, m, \sigma = (\sigma_1, \ldots, \sigma_n))$: check that $f(\sigma_i) = v_i^{m_i}$ for all $i \in [n]$

OWF $\implies$ Length Restricted, One Time Signature

Construction 22 (length-restricted, one-time signature)

Let $f\colon \{0,1\}^n \mapsto \{0,1\}^n$.

1. $\mathsf{Gen}(1^n)$:
   1. $s_1^0, s_1^1, \ldots, s_n^0, s_n^1 \leftarrow \{0,1\}^n$,
   2. $s = \left(s_1^0, s_1^1, \ldots, s_n^0, s_n^1\right)$
   3. $v = \left(v_1^0 = f(s_1^0), v_1^1 = f(s_1^1), \ldots, v_n^0 = f(s_n^0), v_n^1 = f(s_n^1)\right)$
2. $\mathsf{Sign}(s, m)$: $\sigma = (s_1^{m_1}, \ldots, s_n^{m_n})$
3. $\mathsf{Vrfy}(v, m, \sigma = (\sigma_1, \ldots, \sigma_n))$: check that $f(\sigma_i) = v_i^{m_i}$ for all $i \in [n]$

Lemma 23

Assume that f is a OWF, then scheme from Construction 22 is a length restricted one-time signature scheme

Proving Lemma 23

Let a PPT A, $\mathcal{I} \subseteq \mathbb{N}$ and $p \in$ poly that break the security of Construction 22, we use A to invert f.

Proving Lemma 23

Let a PPT A, $\mathcal{I} \subseteq \mathbb{N}$ and $p \in$ poly that break the security of Construction 22, we use A to invert f.

Algorithm 24 (Inv)

Input: $y \in \{0,1\}^n$

1. Choose $(s, v) \leftarrow Gen(1^n)$ and replace $v_{j^*}^{i^*}$ for a random $i^* \in [n]$ and $j^* \in \{0,1\}$, with y.
2. If $A(1^n, v)$ asks to sign message $m \in \{0,1\}^n$ with $m_{i^*} = j^*$ abort. Otherwise, use s to answer the query.
3. Let (m, σ) be A's output.
   If σ is not a valid signature for m, or $m_{i^*} \neq j^*$, abort.
   Otherwise, return σ_{i^*}.

Proving Lemma 23

Let a PPT A, $\mathcal{I} \subseteq \mathbb{N}$ and $p \in$ poly that break the security of Construction 22, we use A to invert f.

Algorithm 24 (Inv)

Input: $y \in \{0,1\}^n$

1. Choose $(s, v) \leftarrow Gen(1^n)$ and replace $v_{j^*}^{i^*}$ for a random $i^* \in [n]$ and $j^* \in \{0,1\}$, with y.
2. If $A(1^n, v)$ asks to sign message $m \in \{0,1\}^n$ with $m_{i^*} = j^*$ abort. Otherwise, use s to answer the query.
3. Let (m, σ) be A's output.
   If σ is not a valid signature for m, or $m_{i^*} \neq j^*$, abort.
   Otherwise, return σ_{i^*}.

- v is distributed as is in the real "signature game"

Proving Lemma 23

Let a PPT A, $\mathcal{I} \subseteq \mathbb{N}$ and $p \in$ poly that break the security of Construction 22, we use A to invert f.

Algorithm 24 (Inv)

Input: $y \in \{0,1\}^n$

1. Choose $(s, v) \leftarrow Gen(1^n)$ and replace $v_{j^*}^{i^*}$ for a random $i^* \in [n]$ and $j^* \in \{0,1\}$, with y.
2. If $A(1^n, v)$ asks to sign message $m \in \{0,1\}^n$ with $m_{i^*} = j^*$ abort. Otherwise, use s to answer the query.
3. Let (m, σ) be A's output.
   If σ is not a valid signature for m, or $m_{i^*} \neq j^*$, abort.
   Otherwise, return σ_{i^*}.

- v is distributed as is in the real "signature game"
- v is independent of i^* and j^*.

Proving Lemma 23

Let a PPT A, $\mathcal{I} \subseteq \mathbb{N}$ and $p \in$ poly that break the security of Construction 22, we use A to invert f.

Algorithm 24 (Inv)

Input: $y \in \{0,1\}^n$

1. Choose $(s, v) \leftarrow Gen(1^n)$ and replace $v_{j^*}^{i^*}$ for a random $i^* \in [n]$ and $j^* \in \{0,1\}$, with y.
2. If $A(1^n, v)$ asks to sign message $m \in \{0,1\}^n$ with $m_{i^*} = j^*$ abort. Otherwise, use s to answer the query.
3. Let (m, σ) be A's output.
   If σ is not a valid signature for m, or $m_{i^*} \neq j^*$, abort.
   Otherwise, return σ_{i^*}.

- v is distributed as is in the real “signature game"
- v is independent of i^* and j^*.
- Therefore Inv inverts f w.p. $\frac{1}{2np(n)}$ for every $n \in \mathcal{I}$.

Stateful schemes (also known as, Memory-dependant schemes)

Definition 25 (Stateful scheme)

Same as in Definition 15, but Sign might keep state.

Stateful schemes (also known as, Memory-dependant schemes)

Definition 25 (Stateful scheme)

Same as in Definition 15, but Sign might keep state.

- Make sense in many applications (e.g., smartcards)

Stateful schemes (also known as, Memory-dependant schemes)

Definition 25 (Stateful scheme)

Same as in Definition 15, but Sign might keep state.

- Make sense in many applications (e.g., smartcards)
- We'll later use it a building block for building stateless scheme

Stateful schemes – Naive Construction

Let $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ be a one-time signature scheme.

Construction 26 (Naive construction)

- $\mathsf{Gen}'(1^n)$: Set $(s_1, v_1) \leftarrow \mathsf{Gen}(1^n)$.
- $\mathsf{Sign}'_{s_1}(m_i)$, where m_i is i'th message to sign:
  1. Let $(s_{i+1}, v_{i+1}) \leftarrow \mathsf{Gen}(1^n)$
  2. Let $\sigma_i = \mathsf{Sign}_{s_i}(m_i, v_{i+1})$
  3. Output $\sigma'_i = (\sigma'_{i-1}, m_i, v_{i+1}, \sigma_i)$.[a]
- $\mathsf{Vrfy}'_{v_1}(m, \sigma' = (m_1, v_2, \sigma_1), \ldots, (m_i, v_{i+1}, \sigma_i))$:
  Check that
  1. $\mathsf{Vrfy}_{v_j}((m_j, v_{j+1}), \sigma_j) = 1$ for every $j \in [i]$
  2. $m_i = m$

[a] σ'_0 is the empty string.

Stateful schemes – Naive Construction

Let $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ be a one-time signature scheme.

Construction 26 (Naive construction)

- $\mathsf{Gen}'(1^n)$: Set $(s_1, v_1) \leftarrow \mathsf{Gen}(1^n)$.
- $\mathsf{Sign}'_{s_1}(m_i)$, where m_i is i'th message to sign:
  1. Let $(s_{i+1}, v_{i+1}) \leftarrow \mathsf{Gen}(1^n)$
  2. Let $\sigma_i = \mathsf{Sign}_{s_i}(m_i, v_{i+1})$
  3. Output $\sigma'_i = (\sigma'_{i-1}, m_i, v_{i+1}, \sigma_i)$.[a]
- $\mathsf{Vrfy}'_{v_1}(m, \sigma' = (m_1, v_2, \sigma_1), \ldots, (m_i, v_{i+1}, \sigma_i))$:
  Check that
  1. $\mathsf{Vrfy}_{v_j}((m_j, v_{j+1}), \sigma_j) = 1$ for every $j \in [i]$
  2. $m_i = m$

[a] σ'_0 is the empty string.

We sometimes refer to (s_1, v_1) generated by Gen above as (s', v')

Naive Construction cont.

- The state of Sign' is used for maintaining the recently used private key (e.g., s_i) and to prevent from using the same one-time signature twice.

Naive Construction cont.

- The state of Sign' is used for maintaining the recently used private key (e.g., s_i) and to prevent from using the same one-time signature twice.
- Inefficient scheme, thought still polynomial, both running time and signature size are linear in number of signatures

Naive Construction cont.

- The state of Sign' is used for maintaining the recently used private key (e.g., s_i) and to prevent from using the same one-time signature twice.
- Inefficient scheme, thought still polynomial, both running time and signature size are linear in number of signatures
- Uses the fact that $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ works for any length (specifically, it is possible to sign message that is longer than the verification key)

Lemma 27

Assume that $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ *is one time signature scheme, then* $(\mathsf{Gen}', \mathsf{Sign}', \mathsf{Vrfy}')$ *is a stateful existential unforgeable signature scheme.*

Lemma 27

Assume that $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ *is one time signature scheme, then* $(\mathsf{Gen}', \mathsf{Sign}', \mathsf{Vrfy}')$ *is a stateful existential unforgeable signature scheme.*

Proof: Let A' be a PPT that breaks the security of $(\mathsf{Gen}', \mathsf{Sign}', \mathsf{Vrfy}')$ with respect to $\mathcal{I} \subseteq \mathbb{N}$ and $p \in \mathrm{poly}$, we present PPT A that breaks the security of $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$.

Lemma 27

Assume that $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ *is one time signature scheme, then* $(\mathsf{Gen}', \mathsf{Sign}', \mathsf{Vrfy}')$ *is a stateful existential unforgeable signature scheme.*

Proof: Let A' be a PPT that breaks the security of $(\mathsf{Gen}', \mathsf{Sign}', \mathsf{Vrfy}')$ with respect to $\mathcal{I} \subseteq \mathbb{N}$ and $p \in \mathrm{poly}$, we present PPT A that breaks the security of $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$.

- We assume for simplicity that p also bounds the query complexity of A'

Proving Lemma 27 cont.

Let rv $(m_t, \sigma' = (m_1, v_2, \sigma_1), \ldots, (m_t, v_{t+1}, \sigma_t))$ be the pair output by A'

Proving Lemma 27 cont.

Let rv $(m_t, \sigma' = (m_1, v_2, \sigma_1), \ldots, (m_t, v_{t+1}, \sigma_t))$ be the pair output by A′

Claim 28

Whenever A′ succeeds, $\exists \widetilde{i} \in [p]$ such that:

1. Sign' has output $\sigma'_{\widetilde{i}-1} = (m_1, v_2, \sigma_1), \ldots, (m_{\widetilde{i}-1}, v_{\widetilde{i}}, \sigma_{\widetilde{i}-1})$
2. Sign' has not output $\sigma'_{\widetilde{i}} = (m_1, v_2, \sigma_1), \ldots, (m_{\widetilde{i}}, v_{\widetilde{i}+1}, \sigma_{\widetilde{i}})$

Proving Lemma 27 cont.

Let rv $(m_t, \sigma' = (m_1, v_2, \sigma_1), \ldots, (m_t, v_{t+1}, \sigma_t))$ be the pair output by A′

Claim 28

Whenever A′ succeeds, $\exists \widetilde{i} \in [p]$ such that:

1. Sign′ has output $\sigma'_{\widetilde{i}-1} = (m_1, v_2, \sigma_1), \ldots, (m_{\widetilde{i}-1}, v_{\widetilde{i}}, \sigma_{\widetilde{i}-1})$
2. Sign′ has not output $\sigma'_{\widetilde{i}} = (m_1, v_2, \sigma_1), \ldots, (m_{\widetilde{i}}, v_{\widetilde{i}+1}, \sigma_{\widetilde{i}})$

Proof: ?

Proving Lemma 27 cont.

Let rv $(m_t, \sigma' = (m_1, v_2, \sigma_1), \ldots, (m_t, v_{t+1}, \sigma_t))$ be the pair output by A′

Claim 28

Whenever A′ succeeds, $\exists \widetilde{i} \in [p]$ such that:

1. Sign′ has output $\sigma'_{\widetilde{i}-1} = (m_1, v_2, \sigma_1), \ldots, (m_{\widetilde{i}-1}, v_{\widetilde{i}}, \sigma_{\widetilde{i}-1})$
2. Sign′ has not output $\sigma'_{\widetilde{i}} = (m_1, v_2, \sigma_1), \ldots, (m_{\widetilde{i}}, v_{\widetilde{i}+1}, \sigma_{\widetilde{i}})$

Proof: ?

- $v_{\widetilde{i}}$ was sampled by Sign′

Proving Lemma 27 cont.

Let rv $(m_t, \sigma' = (m_1, v_2, \sigma_1), \ldots, (m_t, v_{t+1}, \sigma_t))$ be the pair output by A′

Claim 28

Whenever A′ succeeds, $\exists \widetilde{i} \in [p]$ such that:

1. Sign′ has output $\sigma'_{\widetilde{i}-1} = (m_1, v_2, \sigma_1), \ldots, (m_{\widetilde{i}-1}, v_{\widetilde{i}}, \sigma_{\widetilde{i}-1})$
2. Sign′ has not output $\sigma'_{\widetilde{i}} = (m_1, v_2, \sigma_1), \ldots, (m_{\widetilde{i}}, v_{\widetilde{i}+1}, \sigma_{\widetilde{i}})$

Proof: ?

- $v_{\widetilde{i}}$ was sampled by Sign′
- Let $s_{\widetilde{i}}$ be the signing key generated by Sign′ along with $v_{\widetilde{i}}$, and let $\widetilde{m} = (m_{\widetilde{i}}, v_{\widetilde{i}+1})$

Proving Lemma 27 cont.

Let rv $(m_t, \sigma' = (m_1, v_2, \sigma_1), \ldots, (m_t, v_{t+1}, \sigma_t))$ be the pair output by A'

Claim 28

Whenever A' succeeds, $\exists \widetilde{i} \in [p]$ such that:

1. Sign' has output $\sigma'_{\widetilde{i}-1} = (m_1, v_2, \sigma_1), \ldots, (m_{\widetilde{i}-1}, v_{\widetilde{i}}, \sigma_{\widetilde{i}-1})$
2. Sign' has not output $\sigma'_{\widetilde{i}} = (m_1, v_2, \sigma_1), \ldots, (m_{\widetilde{i}}, v_{\widetilde{i}+1}, \sigma_{\widetilde{i}})$

Proof: ?

- $v_{\widetilde{i}}$ was sampled by Sign'
- Let $s_{\widetilde{i}}$ be the signing key generated by Sign' along with $v_{\widetilde{i}}$, and let $\widetilde{m} = (m_{\widetilde{i}}, v_{\widetilde{i}+1})$
- $\mathsf{Vrfy}_{s_{\widetilde{i}}}(\widetilde{m}, \sigma_{\widetilde{i}}) = 1$

Proving Lemma 27 cont.

Let rv $(m_t, \sigma' = (m_1, v_2, \sigma_1), \ldots, (m_t, v_{t+1}, \sigma_t))$ be the pair output by A'

Claim 28

Whenever A' succeeds, $\exists \widetilde{i} \in [p]$ such that:

1. Sign' has output $\sigma'_{\widetilde{i}-1} = (m_1, v_2, \sigma_1), \ldots, (m_{\widetilde{i}-1}, v_{\widetilde{i}}, \sigma_{\widetilde{i}-1})$
2. Sign' has not output $\sigma'_{\widetilde{i}} = (m_1, v_2, \sigma_1), \ldots, (m_{\widetilde{i}}, v_{\widetilde{i}+1}, \sigma_{\widetilde{i}})$

Proof: ?

- $v_{\widetilde{i}}$ was sampled by Sign'
- Let $s_{\widetilde{i}}$ be the signing key generated by Sign' along with $v_{\widetilde{i}}$, and let $\widetilde{m} = (m_{\widetilde{i}}, v_{\widetilde{i}+1})$
- $\mathsf{Vrfy}_{s_{\widetilde{i}}}(\widetilde{m}, \sigma_{\widetilde{i}}) = 1$
- $\mathsf{Sign}_{s_{\widetilde{i}}}$ was not queried by Sign' on $\widetilde{m}$ and output $\sigma_{\widetilde{i}}$.

Proving Lemma 27 cont.

Let rv $(m_t, \sigma' = (m_1, v_2, \sigma_1), \ldots, (m_t, v_{t+1}, \sigma_t))$ be the pair output by A'

Claim 28

Whenever A' succeeds, $\exists \widetilde{i} \in [p]$ such that:

1. Sign' has output $\sigma'_{\widetilde{i}-1} = (m_1, v_2, \sigma_1), \ldots, (m_{\widetilde{i}-1}, v_{\widetilde{i}}, \sigma_{\widetilde{i}-1})$
2. Sign' has not output $\sigma'_{\widetilde{i}} = (m_1, v_2, \sigma_1), \ldots, (m_{\widetilde{i}}, v_{\widetilde{i}+1}, \sigma_{\widetilde{i}})$

Proof: ?

- $v_{\widetilde{i}}$ was sampled by Sign'
- Let $s_{\widetilde{i}}$ be the signing key generated by Sign' along with $v_{\widetilde{i}}$, and let $\widetilde{m} = (m_{\widetilde{i}}, v_{\widetilde{i}+1})$
- $\mathsf{Vrfy}_{s_{\widetilde{i}}}(\widetilde{m}, \sigma_{\widetilde{i}}) = 1$
- $\mathsf{Sign}_{s_{\widetilde{i}}}$ was not queried by Sign' on $\widetilde{m}$ and output $\sigma_{\widetilde{i}}$.
- $\mathsf{Sign}_{s_{\widetilde{i}}}$ was queried at most once by Sign'

Definition of A

Algorithm 29 (A)

Input: v, 1^n

Oracle: Sign_s

1. Choose $i^* \leftarrow [p = p(n)]$ and $(s', v') \leftarrow \mathsf{Gen}'(1^n)$.
2. Emulate a random execution of $\mathsf{A}'^{\mathsf{Sign}'_{s'}}$ with a single twist:
   - On the i^*'th call to $\mathsf{Sign}'_{s'}$, set $v_{i^*} = v$ (rather then choosing it via Gen)
   - When need to sign using s_{i^*}, use Sign_s.
3. Let $(m, \sigma = (m_1, v_1, \sigma_1), \ldots, (m_p, v_p, \sigma_p)) \leftarrow \mathsf{A}'$
4. Output $((m_{i^*}, v_{i^*}), \sigma_{i^*})$ (abort if $i^* > p$))

Definition of A

Algorithm 29 (A)

Input: v, 1^n
Oracle: Sign_s

1. Choose $i^* \leftarrow [p = p(n)]$ and $(s', v') \leftarrow \mathsf{Gen}'(1^n)$.
2. Emulate a random execution of $\mathsf{A}'^{\mathsf{Sign}'_{s'}}$ with a single twist:
   - On the i^*'th call to $\mathsf{Sign}'_{s'}$, set $v_{i^*} = v$ (rather then choosing it via Gen)
   - When need to sign using s_{i^*}, use Sign_s.
3. Let $(m, \sigma = (m_1, v_1, \sigma_1), \ldots, (m_p, v_p, \sigma_p)) \leftarrow \mathsf{A}'$
4. Output $((m_{i^*}, v_{i^*}), \sigma_{i^*})$ (abort if $i^* > p$))

- The emulated game $\mathsf{A}'^{\mathsf{Sign}'_{s'}}$ has the same distribution as the real game.

Definition of A

Algorithm 29 (A)

Input: v, 1^n
Oracle: Sign_s

1. Choose $i^* \leftarrow [p = p(n)]$ and $(s', v') \leftarrow \mathsf{Gen}'(1^n)$.
2. Emulate a random execution of $\mathsf{A}'^{\mathsf{Sign}'_{s'}}$ with a single twist:
   - On the i^*'th call to $\mathsf{Sign}'_{s'}$, set $v_{i^*} = v$ (rather then choosing it via Gen)
   - When need to sign using s_{i^*}, use Sign_s.
3. Let $(m, \sigma = (m_1, v_1, \sigma_1), \ldots, (m_p, v_p, \sigma_p)) \leftarrow \mathsf{A}'$
4. Output $((m_{i^*}, v_{i^*}), \sigma_{i^*})$ (abort if $i^* > p$))

- The emulated game $\mathsf{A}'^{\mathsf{Sign}'_{s'}}$ has the same distribution as the real game.
- Sign_s is called at most once

Definition of A

Algorithm 29 (A)

Input: v, 1^n

Oracle: Sign_s

1. Choose $i^* \leftarrow [p = p(n)]$ and $(s', v') \leftarrow \mathsf{Gen}'(1^n)$.
2. Emulate a random execution of $\mathsf{A}'^{\mathsf{Sign}'_{s'}}$ with a single twist:
   - On the i^*'th call to $\mathsf{Sign}'_{s'}$, set $v_{i^*} = v$ (rather then choosing it via Gen)
   - When need to sign using s_{i^*}, use Sign_s.
3. Let $(m, \sigma = (m_1, v_1, \sigma_1), \ldots, (m_p, v_p, \sigma_p)) \leftarrow \mathsf{A}'$
4. Output $((m_{i^*}, v_{i^*}), \sigma_{i^*})$ (abort if $i^* > p$))

- The emulated game $\mathsf{A}'^{\mathsf{Sign}'_{s'}}$ has the same distribution as the real game.
- Sign_s is called at most once
- A breaks $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ whenever $i^* = \widetilde{i}$.

A "Somewhat"-stateful Scheme

A one-time scheme $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$

Construction 30 (A "Somewhat"-stateful Scheme)

- $\mathsf{Gen}'(1^n)$: Set $(s_\lambda, v_\lambda) \leftarrow \mathsf{Gen}(1^n)$.
- $\mathsf{Sign}'_s(m)$: choose unused $\bar{r} \in \{0,1\}^n$
  1. For $i = 0$ to $n-1$: if $a_{\bar{r}_{1,\ldots,i}}$ was not set before:
     1. For both $j \in \{0,1\}$, let $(s_{\bar{r}_{1,\ldots,i},j}, v_{\bar{r}_{1,\ldots,i},j}) \leftarrow \mathsf{Gen}(1^n)$
     2. Let $\sigma_{\bar{r}_{1,\ldots,i}} = \mathsf{Sign}_{s_{\bar{r}_{1,\ldots,i}}}(a_{\bar{r}_{1,\ldots,i}} = (v_{\bar{r}_{1,\ldots,i},0}, v_{\bar{r}_{1,\ldots,i},1}))$
  2. Output $(\bar{r}, a_\lambda, \sigma_\lambda, \ldots, a_{\bar{r}_{1,\ldots,n-1}}, \sigma_{\bar{r}_{1,\ldots,n-1}}, \sigma_{\bar{r}} = \mathsf{Sign}_{s_{\bar{r}}}(m))$
- $\mathsf{Vrfy}'_{v_\lambda}(m, \sigma' = (\bar{r}, a_\lambda, \sigma_\lambda, \ldots, a_{\bar{r}-1}, \sigma_{\bar{r}_{1,\ldots,n-1}}, \sigma_{\bar{r}})$
  Check that
  1. $\mathsf{Vrfy}_{v_{\bar{r}_{1,\ldots,i}}}(a_{\bar{r}_{1,\ldots,i}}, \sigma_{\bar{r}_{1,\ldots,i}}) = 1$ for every $i \in \{0,\ldots,n-1\}$
  2. $\mathsf{Vrfy}_{v_{\bar{r}}}(m, \sigma_{\bar{r}}) = 1$, for $v_{\bar{r}} = (a_{\bar{r}})_{\bar{r}[n]}$

A "Somewhat"-stateful Scheme, cont.

- More efficient scheme — Enough to construct tree of depth $\omega(\log n)$ (i.e., to choose $\bar{r} \in \{0,1\}^{\ell \in \omega(\log n)}$)

A “Somewhat"-stateful Scheme, cont.

- More efficient scheme — Enough to construct tree of depth $\omega(\log n)$ (i.e., to choose $\bar{r} \in \{0,1\}^{\ell \in \omega(\log n)}$)
- Sign' does not keep track of the message history.

A “Somewhat"-stateful Scheme, cont.

- More efficient scheme — Enough to construct tree of depth $\omega(\log n)$ (i.e., to choose $\bar{r} \in \{0,1\}^{\ell \in \omega(\log n)}$)
- Sign' does not keep track of the message history.
- Each leaf is visited at most once.

A “Somewhat"-stateful Scheme, cont.

- More efficient scheme — Enough to construct tree of depth $\omega(\log n)$ (i.e., to choose $\bar{r} \in \{0,1\}^{\ell \in \omega(\log n)}$)
- Sign' does not keep track of the message history.
- Each leaf is visited at most once.
- Each one-time signature is used once.

Lemma 31

Assume that $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ *is one time signature scheme, then* $(\mathsf{Gen}', \mathsf{Sign}', \mathsf{Vrfy}')$ *is a stateful existential unforgeable signature scheme.*

Lemma 31

Assume that $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ *is one time signature scheme, then* $(\mathsf{Gen}', \mathsf{Sign}', \mathsf{Vrfy}')$ *is a stateful existential unforgeable signature scheme.*

Proof: ?

Stateless Scheme

Let Π_n be the set of all functions from $\bigcup_{i=1}^{n}\{0,1\}^i$ to $\{0,1\}^{q(n)}$ for some "large enough" $q \in \text{poly}$ and let $\mathcal{H}$ be a CRH.

Construction 32 (Inefficient stateless Scheme)

- $\text{Gen}'(1^n)$: Set $(s_\lambda, v_\lambda) \leftarrow \text{Gen}(1^n)$ and $\pi \leftarrow \Pi_{q(n)}$ and $h \leftarrow \mathcal{H}_n$, and output $(s' = (s, \pi, h), v' = v)$

Stateless Scheme

Let Π_n be the set of all functions from $\bigcup_{i=1}^{n} \{0,1\}^i$ to $\{0,1\}^{q(n)}$ for some "large enough" $q \in \mathrm{poly}$ and let $\mathcal{H}$ be a CRH.

Construction 32 (Inefficient stateless Scheme)

- $\mathrm{Gen}'(1^n)$: Set $(s_\lambda, v_\lambda) \leftarrow \mathrm{Gen}(1^n)$ and $\pi \leftarrow \Pi_{q(n)}$ and $h \leftarrow \mathcal{H}_n$, and output $(s' = (s, \pi, h), v' = v)$
- $\mathrm{Sign}'_s(m)$: choose $\bar{r} = \pi(h(m))_{1,\ldots,n}$.
  1. For $i = 0$ to $n-1$: if $a_{\bar{r}_{1,\ldots,i}}$ was not set before:
     1. For both $j \in \{0,1\}$, let $(s_{\bar{r}_{1,\ldots,i},j}, v_{\bar{r}_{1,\ldots,i},j}) \leftarrow \mathrm{Gen}(1^n; \pi(\bar{r}_{1,\ldots,i}, j))$
     2. Let $\sigma_{\bar{r}_{1,\ldots,i}} = \mathrm{Sign}_{s_{\bar{r}_{1,\ldots,i}}}(a_{\bar{r}_{1,\ldots,i}} = (v_{\bar{r}_{1,\ldots,i},0}, v_{\bar{r}_{1,\ldots,i},1}))$
  2. Output $(\bar{r}, a_\lambda, \sigma_\lambda, \ldots, a_{\bar{r}_{1,\ldots,n-1}}, \sigma_{\bar{r}_{1,\ldots,n-1}}, \sigma_{\bar{r}} = \mathrm{Sign}_{s_{\bar{r}}}(m))$
- Vrfy': unchanged

Stateless Scheme

Let Π_n be the set of all functions from $\bigcup_{i=1}^{n}\{0,1\}^i$ to $\{0,1\}^{q(n)}$ for some "large enough" $q \in \text{poly}$ and let $\mathcal{H}$ be a CRH.

Construction 32 (Inefficient stateless Scheme)

- $\text{Gen}'(1^n)$: Set $(s_\lambda, v_\lambda) \leftarrow \text{Gen}(1^n)$ and $\pi \leftarrow \Pi_{q(n)}$ and $h \leftarrow \mathcal{H}_n$, and output $(s' = (s, \pi, h), v' = v)$
- $\text{Sign}'_s(m)$: choose $\bar{r} = \pi(h(m))_{1,\ldots,n}$.
  1. For $i = 0$ to $n-1$: if $a_{\bar{r}_{1,\ldots,i}}$ was not set before:
     1. For both $j \in \{0,1\}$, let $(s_{\bar{r}_{1,\ldots,i},j}, v_{\bar{r}_{1,\ldots,i},j}) \leftarrow \text{Gen}(1^n; \pi(\bar{r}_{1,\ldots,i}, j))$
     2. Let $\sigma_{\bar{r}_{1,\ldots,i}} = \text{Sign}_{s_{\bar{r}_{1,\ldots,i}}}(a_{\bar{r}_{1,\ldots,i}} = (v_{\bar{r}_{1,\ldots,i},0}, v_{\bar{r}_{1,\ldots,i},1}))$
  2. Output $(\bar{r}, a_\lambda, \sigma_\lambda, \ldots, a_{\bar{r}_{1,\ldots,n-1}}, \sigma_{\bar{r}_{1,\ldots,n-1}}, \sigma_{\bar{r}} = \text{Sign}_{s_{\bar{r}}}(m))$
- Vrfy': unchanged

- A single one-time signature key might be used several times, but always on the same message

Stateless Scheme

Let Π_n be the set of all functions from $\bigcup_{i=1}^{n}\{0,1\}^i$ to $\{0,1\}^{q(n)}$ for some "large enough" $q \in \mathrm{poly}$ and let $\mathcal{H}$ be a CRH.

Construction 32 (Inefficient stateless Scheme)

- $\mathsf{Gen}'(1^n)$: Set $(s_\lambda, v_\lambda) \leftarrow \mathsf{Gen}(1^n)$ and $\pi \leftarrow \Pi_{q(n)}$ and $h \leftarrow \mathcal{H}_n$, and output $(s' = (s, \pi, h), v' = v)$
- $\mathsf{Sign}'_s(m)$: choose $\overline{r} = \pi(h(m))_{1,\ldots,n}$.
  1. For $i = 0$ to $n-1$: if $a_{\overline{r}_{1,\ldots,i}}$ was not set before:
     1. For both $j \in \{0,1\}$, let $(s_{\overline{r}_{1,\ldots,i},j}, v_{\overline{r}_{1,\ldots,i},j}) \leftarrow \mathsf{Gen}(1^n; \pi(\overline{r}_{1,\ldots,i}, j))$
     2. Let $\sigma_{\overline{r}_{1,\ldots,i}} = \mathsf{Sign}_{s_{\overline{r}_{1,\ldots,i}}}(a_{\overline{r}_{1,\ldots,i}} = (v_{\overline{r}_{1,\ldots,i},0}, v_{\overline{r}_{1,\ldots,i},1}))$
  2. Output $(\overline{r}, a_\lambda, \sigma_\lambda, \ldots, a_{\overline{r}_{1,\ldots,n-1}}, \sigma_{\overline{r}_{1,\ldots,n-1}}, \sigma_{\overline{r}} = \mathsf{Sign}_{s_{\overline{r}}}(m))$
- Vrfy': unchanged

- A single one-time signature key might be used several times, but always on the same message

Stateless Scheme

Let Π_n be the set of all functions from $\bigcup_{i=1}^{n}\{0,1\}^i$ to $\{0,1\}^{q(n)}$ for some "large enough" $q \in \text{poly}$ and let $\mathcal{H}$ be a CRH.

Construction 32 (Inefficient stateless Scheme)

- $\mathsf{Gen}'(1^n)$: Set $(s_\lambda, v_\lambda) \leftarrow \mathsf{Gen}(1^n)$ and $\pi \leftarrow \Pi_{q(n)}$ and $h \leftarrow \mathcal{H}_n$, and output $(s' = (s, \pi, h), v' = v)$
- $\mathsf{Sign}'_s(m)$: choose $\bar{r} = \pi(h(m))_{1,\ldots,n}$.
  1. For $i = 0$ to $n-1$: if $a_{\bar{r}_{1,\ldots,i}}$ was not set before:
     1. For both $j \in \{0,1\}$, let $(s_{\bar{r}_{1,\ldots,i},j}, v_{\bar{r}_{1,\ldots,i},j}) \leftarrow \mathsf{Gen}(1^n; \pi(\bar{r}_{1,\ldots,i}, j))$
     2. Let $\sigma_{\bar{r}_{1,\ldots,i}} = \mathsf{Sign}_{s_{\bar{r}_{1,\ldots,i}}}(a_{\bar{r}_{1,\ldots,i}} = (v_{\bar{r}_{1,\ldots,i},0}, v_{\bar{r}_{1,\ldots,i},1}))$
  2. Output $(\bar{r}, a_\lambda, \sigma_\lambda, \ldots, a_{\bar{r}_{1,\ldots,n-1}}, \sigma_{\bar{r}_{1,\ldots,n-1}}, \sigma_{\bar{r}} = \mathsf{Sign}_{s_{\bar{r}}}(m))$
- Vrfy': unchanged

- A single one-time signature key might be used several times, but always on the same message

Efficient scheme:

Stateless Scheme

Let Π_n be the set of all functions from $\bigcup_{i=1}^{n}\{0,1\}^i$ to $\{0,1\}^{q(n)}$ for some "large enough" $q \in \text{poly}$ and let $\mathcal{H}$ be a CRH.

Construction 32 (Inefficient stateless Scheme)

- $\text{Gen}'(1^n)$: Set $(s_\lambda, v_\lambda) \leftarrow \text{Gen}(1^n)$ and $\pi \leftarrow \Pi_{q(n)}$ and $h \leftarrow \mathcal{H}_n$, and output $(s' = (s, \pi, h), v' = v)$
- $\text{Sign}'_s(m)$: choose $\bar{r} = \pi(h(m))_{1,\ldots,n}$.
  1. For $i = 0$ to $n-1$: if $a_{\bar{r}_{1,\ldots,i}}$ was not set before:
     1. For both $j \in \{0,1\}$, let $(s_{\bar{r}_{1,\ldots,i},j}, v_{\bar{r}_{1,\ldots,i},j}) \leftarrow \text{Gen}(1^n; \pi(\bar{r}_{1,\ldots,i}, j))$
     2. Let $\sigma_{\bar{r}_{1,\ldots,i}} = \text{Sign}_{s_{\bar{r}_{1,\ldots,i}}}(a_{\bar{r}_{1,\ldots,i}} = (v_{\bar{r}_{1,\ldots,i},0}, v_{\bar{r}_{1,\ldots,i},1}))$
  2. Output $(\bar{r}, a_\lambda, \sigma_\lambda, \ldots, a_{\bar{r}_{1,\ldots,n-1}}, \sigma_{\bar{r}_{1,\ldots,n-1}}, \sigma_{\bar{r}} = \text{Sign}_{s_{\bar{r}}}(m))$
- Vrfy': unchanged

- A single one-time signature key might be used several times, but always on the same message

Efficient scheme: use PRF

Getting rid of the CRH

Getting rid of the CRH

Definition 33 (target collision resistant (TCR))

A function family $\mathcal{H} = \{\mathcal{H}_n\}$ is target collision resistant, if any pair of PPT's $\mathsf{A}_1, \mathsf{A}_2$:

$$\Pr[(x, a) \leftarrow \mathsf{A}_1(1^n); h \leftarrow \mathcal{H}_n; x' \leftarrow \mathsf{A}_2(a, h) : \\ x \neq x' \wedge h(x) = h(x')] = \operatorname{neg}(n)$$

Getting rid of the CRH

Definition 33 (target collision resistant (TCR))

A function family $\mathcal{H} = \{\mathcal{H}_n\}$ is target collision resistant, if any pair of PPT's $\mathsf{A}_1, \mathsf{A}_2$:

$$\Pr[(x,a) \leftarrow \mathsf{A}_1(1^n); h \leftarrow \mathcal{H}_n; x' \leftarrow \mathsf{A}_2(a,h) : x \neq x' \wedge h(x) = h(x')] = \operatorname{neg}(n)$$

Theorem 34

OWFs imply efficient compressing TCRs.

Definition 35 (target one-time signatures)

A signature scheme $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ is target one-time existential unforgeable (for short, target one-time signature), if for any pair of PPT's $\mathsf{A}_1, \mathsf{A}_2$

$$\Pr\big[(m, a) \leftarrow \mathsf{A}_1(1^n); (s, v) \leftarrow \mathsf{Gen}(1^n);$$
$$(m', \sigma) \leftarrow \mathsf{A}(a, \mathsf{Sign}_s(m))\colon m' \neq m \wedge \mathsf{Vrfy}_v(m', \sigma) = 1\big]$$
$$= \mathsf{neg}(n)$$

Definition 35 (target one-time signatures)

A signature scheme $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ is target one-time existential unforgeable (for short, target one-time signature), if for any pair of PPT's $\mathsf{A}_1, \mathsf{A}_2$

$$\Pr\big[(m,a) \leftarrow \mathsf{A}_1(1^n); (s,v) \leftarrow \mathsf{Gen}(1^n);$$
$$(m',\sigma) \leftarrow \mathsf{A}(a, \mathsf{Sign}_s(m)) \colon m' \neq m \wedge \mathsf{Vrfy}_v(m',\sigma) = 1\big]$$
$$= \mathsf{neg}(n)$$

Claim 36

OWFs imply target one-time signatures.

Definition 37 (random one-time signatures)

A signature scheme $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ is random one-time existential unforgeable (for short, random one-time signature), if for any PPT A and any samplable ensemble $\mathcal{M} = \{\mathcal{M}_n\}_{n\in\mathbb{N}}$, it holds that

$$\begin{aligned}&\Pr\big[m \leftarrow \mathcal{M}_n; (s, v) \leftarrow \mathsf{Gen}(1^n); (m', \sigma) \leftarrow \mathsf{A}(m, \mathsf{Sign}_s(m)) :\\ &\quad m' \neq m \wedge \mathsf{Vrfy}_v(m', \sigma) = 1\big]\\ &\quad = \mathsf{neg}(n)\end{aligned}$$

Definition 37 (random one-time signatures)

A signature scheme $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ is random one-time existential unforgeable (for short, random one-time signature), if for any PPT A and any samplable ensemble $\mathcal{M} = \{\mathcal{M}_n\}_{n\in\mathbb{N}}$, it holds that

$$\Pr\big[m \leftarrow \mathcal{M}_n; (s,v) \leftarrow \mathsf{Gen}(1^n); (m',\sigma) \leftarrow \mathsf{A}(m, \mathsf{Sign}_s(m)) : m' \neq m \wedge \mathsf{Vrfy}_v(m',\sigma) = 1\big] = \mathsf{neg}(n)$$

Claim 38

Assume $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ is target one-time existential unforgeable, then it is random one-time existential unforgeable.

Lemma 39

Assume that the underlying one-time signature scheme is target one-time and the hash function (e.g., $\mathcal{H}$) is a TCR, then Construction 32 is existential unforgeable signature scheme.

Lemma 39

Assume that the underlying one-time signature scheme is target one-time and the hash function (e.g., $\mathcal{H}$) is a TCR, then Construction 32 is existential unforgeable signature scheme.

Proof:

Lemma 39

Assume that the underlying one-time signature scheme is target one-time and the hash function (e.g., $\mathcal{H}$) is a TCR, then Construction 32 is existential unforgeable signature scheme.

Proof:

- Prove that if the underlying signature scheme is target one-time, then Construction 30 is stateful existential unforgeable

Lemma 39

Assume that the underlying one-time signature scheme is target one-time and the hash function (e.g., $\mathcal{H}$) is a TCR, then Construction 32 is existential unforgeable signature scheme.

Proof:

- Prove that if the underlying signature scheme is target one-time, then Construction 30 is stateful existential unforgeable
- Prove that Construction 32 when used with a CRH is existential unforgeable signature scheme

Lemma 39

Assume that the underlying one-time signature scheme is target one-time and the hash function (e.g., $\mathcal{H}$) is a TCR, then Construction 32 is existential unforgeable signature scheme.

Proof:

- Prove that if the underlying signature scheme is target one-time, then Construction 30 is stateful existential unforgeable
- Prove that Construction 32 when used with a CRH is existential unforgeable signature scheme
- Show that the underlying CRH can be safely replaced with a TCR